AJEMA GLOBAL RESEARCH NETWORK

AMERICAN Journal of Engineering, Mechanics and Architecture
Volume 1, Issue 04, 2023 ISSN (E): 2993-2637

Persistent Memory Poisoning in Long-Lived AI Agents: A Threat Analysis and Trust-Aware Memory Defense Framework

Gaurav Chaulagain[1], Medhavee Upadhyaya[2]

[1] *Department of Electrical and Computer Engineering, San Francisco Bay University*

gchaulag977@student.sfbu.edu

[2] *Department of Electrical and Computer Engineering, San Francisco Bay University*

mupadhya200@student.sfbu.edu

Abstract: Many Artificial Intelligent Model (AI) based agents are using the persistent memory for maintaining the context information, remembering the user preferences and streamlining the performance of the tasks in multiple interactions with the user. This capability enables long-term personalization, but also creates a security risk because the agent's behavior could be affected by information from another interaction, albeit from an adversarial source. Persistent memory poisoning can be a more comprehensive form of attack in which the malicious information can still have a significant effect even though the original source is no longer in existence. In this paper, persistent memory poisoning in long-lived AI agents is explored, and a trust-aware defense framework called M-Guard is introduced, which incorporates provenance tracking, memory-write validation, trust-aware retrieval, temporal revalidation, and execution-time verification. The proposed research methodology is based on a series of controlled scenarios that include direct memory injection, misleading historical information, multi-record contamination, and delayed trigger-based behavior. A simulation-based evaluation is intended to compare the performance of the following: unrestricted memory, prompt-level filtering, write-time validation, retrieval-time screening, and the full system. The simulated "placeholders" in the example results provided in this draft are not empirical results. They show how to systematically measure the effectiveness of security as well as the level of security that can be achieved in completing a legitimate task, the false-positive rate, and how much memory is retained. It is the study's contention that persistent memory must be considered as being in a state of security, not automatically assumed to be a trusted repository of knowledge. The proposed framework provides a basis for reproducible experimentation and subsequent validation with real AI agents.

Keywords: Persistent memory poisoning, artificial intelligent Model agents, AI security, long-term memory, prompt injection, trust-aware retrieval, provenance tracking, memory integrity.

Introduction

A. Background

Artificial Intelligence (AI) systems are becoming more and more capable of handling information, finding relevant knowledge and retaining helpful context between tasks. Since the beginnings of machine learning with neural networks, memories have been explored as a way of

augmenting the capabilities of a network with information not in its immediate input. Memory networks are used to store and retrieve information [1] and differentiable neural computers are those that use an external memory structure with neural computation [2] are examples of such systems. Lewis et al. presented a parametric framework with retrieved knowledge, which is a non-parametric approach [3]. Guu et al. introduced REALM, a dense retrieval model jointly trained with language-model pretraining [4] and Karpukhin et al. showed the effectiveness of dense retrieval for open-domain question answering [5]. Following studies explored retrieval-based generation and enhanced ways of leveraging retrieved passages to answer questions [6][7]. The methods presented here highlight that external information can enhance the capabilities of AI systems, but also that it introduces a requirement for reliable and accurate information in storage. Thus, the security of such information is important to consider. Adversarial Machine Learning has shown how a model can be influenced by maliciously inputted data or by corrupting its training data [8][9][10]. Another example is backdoor attacks: An attacker may add an association that makes a model misbehave when a certain trigger is shown [11][12]. While the main focus of both of these studies lies in the training of the model or input manipulation, they do get the general idea that an AI system can create unwanted behavior when the inputs or processes upon which it depends upon have been manipulated.

B. Problem Statement

Persistent Memory poisoning is an attack where the adversary feeds misleading, false, or malicious information into an AI agent's long-term memory which can impact its later responses or actions. The effect can be a single interaction, but can persist beyond the view of the initial content in the active conversation. This results in a potential security problem for systems that store information from previous tasks and make decisions based on that information in the future. Previous studies offer a number of bases to appreciate this menace. The research on data poisoning focuses on the impact of tampered training instances on learning and prediction [9]. In backdoor research, it's shown that malicious associations can be hidden for a specific trigger [11][12]. Another study of adversarial examples in NLP also demonstrates that changes in the input can affect the output of an NLP model [13][14]. These research areas, however, alone do not provide a complete understanding of how untrusted information can be stored, retrieved, transformed, and reused through multiple interactions with the memory-enabled AI agent. The issue isn't just about inserting a bad record. It also involves the possibility of the record being retrieved later, whether the agent uses the record as evidence, and the impact on a later decision. Requiring a defense to consider only the initial input can fail to consider risks after information is stored. Likewise, a memory system that does not remember the source of the information could make it harder to tell the truth from unverifiable or conflicting information. The objective of this study is thus to examine Persistent Memory Poisoning as a lifecycle security issue. It suggests a framework, which we call M-Guard (Memory Guard with Provenance, Validation and Temporal Reassessment), to be used to assess memory records at multiple stages and minimize the chances that the malicious information will impact the subsequent agent action.

C. Research Objectives

The objectives of this study are:

1. To describe the attack phases and threat model of persistent memory poisoning.
2. Design and build a memory defense mechanism that is trust aware for long-lived AI agents.
3. To develop an experimental methodology to assess memory poisoning for several sessions and attack scenarios.
4. To evaluate the proposed framework with simpler protection mechanisms using security

and utility metrics.

5. To help determine any remaining weaknesses, obstacles, and avenues for further study.

D. Research Questions

The study aims to answer the following research questions:

RQ1: What are the effects of persistent memory poisoning on the behaviour of an agent in multiple sessions versus attacks that occur only in one session?

RQ2: How effective is provenance tracking, write-time validation and trust-aware retrieval in decreasing the success of memory poisoning attacks?

RQ3: Is there a tradeoff between malicious behavior and maintaining task completion?

RQ4: What kind of attacks are still hard to detect when malicious information is stored in multiple memory records and is triggered later by different contextual stimuli?

E. Research Contributions

There are four planned contributions.

Firstly, this paper provides a threat model based on the life cycle of the paper, i.e., paper creation, storage, retrieval, use, and removal.

Secondly, it suggests a layered model called M-Guard which associates memory records with their origin and which evaluates them before storing and using them.

Third, it establishes a measurable, repeatable design for evaluating the success of the attack, how long the poisoned memory remains active, the detection performance, and the success of legitimate tasks.

Lastly, it offers a formal experimental analysis, where residual risks such as trusted source confusion, multi-record interactions and delayed activation are made explicit.

Related Work and Research Gap

A. Memory Architectures and Retrieval-Augmented Systems

The study of memory mechanisms can help to enhance the capacity of neural systems to remember and recall information. The Weston et al. proposal of memory networks based on explicit memory to support reasoning over memory facts [15] introduced this idea. Graves et al. introduced the concept of differentiable neural computers, a hybrid of neural networks and an external memory that can be read and written during the computation [2]. These methods introduced key ideas for systems that rely on information stored in the system, and not just on the immediate information they receive. An related basis is retrieval-augmented language processing. Lewis et al. used a pretrained sequence-to-sequence model and a dense vector index of external documents [3]. The knowledge retrieval for language-model pretraining is integrated into REALM [4] and Dense Passage Retrieval is used to improve retrieval for open domain question answering [5]. Following that, Izacard and Grave investigated the possibility of using passages from a retrieved document as a basis for generating answers to a question [6]. Borgeaud et al. showed that accessing data from an external large data store can boost the performance of a language model [7]. Studies on these aspects are mainly on capability, retrieval quality and

answer generation. What they found is applicable to persistent memory security as it demonstrates how much downstream output relies on the retrieved information. But just because content is retrieved doesn't necessarily mean the content is accurate and reliable. If there is a malicious record that is relevant to a search, then a retrieval system successfully retrieves that record. This is an incentive to study memory integrity as well as retrieval performance, along with memory trust assessment.

B. Data Poisoning and Backdoor Attacks

Data poisoning is a well-studied problem in the field of adversarial machine learning. Biggio et al. presented a study on poisoning attacks against support vector machines (SVMs), demonstrating that an attacker can alter training data to alter the decision boundaries learned by the machine [8]. Steinhardt et al. explored the shortcomings of defenses against data poisoning and come up with a way to reason about data poisoning robustness [9]. In an attempt to find methods of identifying influential training examples, Koh and Liang studied techniques that can be used to analyse how individual data points impact on the model prediction [10]. Another type of persistent malicious influence is backdoor attacks. Gu et al. [11] proposed a technique of training a model to generate an attacker-specified output after receiving a trigger, called BadNets. In [12] Chen et al. investigated the targeted backdoor attack method on deep neural networks and examined the effects of poisonings to induce attacker-controlled behavior. These works illustrate the fact that the harmful effects can only manifest under specific circumstances and conditions, but not normally. Adversarial manipulation is also possible for natural-language systems. Ebrahimi et al. proposed attacks at character level which alter the text input to the model to change its prediction [13]. Wallace et al. studied universal adversarial triggers, demonstrating that certain sequences of tokens can lead to the expression of undesirable behaviors under inputs [14]. Kurita et al. studied the poisoning of pre-trained language models to add specific behaviours [15]. While these studies offer concepts that can be useful to consider when dealing with malicious data and hidden triggers, they have taken a different approach to the persistent memory poisoning focus. Attacks can be done by traditional training-time poisoning (TT), which can alter the examples used to train a model, or by memory-poisoning (MP), which can corrupt information stored during normal operation. The model can be the same and later actions can be affected by the tainted memory. This difference leads to a framework to evaluate memory as a distinct and possibly vulnerable aspect.

C. Security and Reliability of Language Models

Studies on the trustworthiness of language models reveal other issues with how AI systems behave. In natural language processing, Wallace et al. explored adversarial examples, showing that models can be sensitive to text inputs that are carefully crafted [14]. Kurita et al. demonstrated that language models can be sensitive to adversarial poisoning attacks [15]. The studies suggest that system reliability is related to not only the architecture of the system, but also the information that is fed into and stored by the system. Bender et al. explored limitations of large language models: impact of training data composition, bias, and reliable interpretations of generated language [16]. Weidinger et al. created a taxonomy of dangers related to language models (LMs), such as dangers linked to malicious use, discrimination, misinformation and human-computer interaction (HCI) [17]. The analyses presented here reinforce the general discussion regarding the risk of AI-systems, which has been presented from a different perspective than just model accuracy. Other security frameworks are also valuable starting points. NIST's approach to enhancing the cybersecurity of critical infrastructure provides a framework of (identify, protect, detect, respond, and recover) functions [18]. While not designed specifically for AI-agent memory, the risk-management approach could be used to inform the organization of memory protection controls. Adversarial Machine Learning (AML) also provides a foundation that the attacker's capabilities, access, objectives and assumptions must all be explicitly defined when assessing the defender [19][20]. These principles indicate that for

persistent memory systems, the source of the information to be stored should be taken into account, the points at which an attacker can manipulate the information and the effects of manipulating the information later. Unintended effects, such as denial of valid information or lowering normal task performance should also be considered in a defense.

D. Memory Integrity, Provenance, and Trust-Aware Defenses

Memory Integrity is the accuracy, authorization and attribution of stored information. Retrieval-augmented systems are reliant on external documents, and memory-based systems can be as well, and can be reliant on information from past interactions. When a source for a record is not identified when copying, summarizing, or transforming, subsequent parts of a record might have problems determining that it is reliable. The aim of data sanitization is to detect and remove suspicious examples before they affect learning [21]. Influence analysis can be used to explore the influence of individual training examples [22]. The significance of using backdoor research, to evaluate behavior under both ordinary and trigger conditions, can be seen in [23][24]. The following ideas can be translated conceptually to the memory lifecycle, but more experimental testing is needed to validate their utility in persistent AI-agent memory.

This study proposes M-Guard as a layered approach comprising five elements:

1. **Provenance-aware records:** Retain information about the source and origin of each memory record.

2. **Write-time validation:** Examine candidate records before accepting them into persistent memory.

3. **Trust-aware retrieval:** Evaluate a record's trust status in addition to its relevance to the current task.

4. **Temporal reassessment:** Reconsider the validity and trustworthiness of stored information as circumstances change.

5. **Execution-time verification:** Check whether actions influenced by retrieved memories are authorized and consistent with the task's constraints.

These components are proposed design choices, not established outcomes from the cited studies. Their effectiveness must be assessed experimentally, including whether they reduce malicious behavior without excessively blocking legitimate memory use.

E. Research Gap

The survey above provides a number of bases for the study of persistent memory poisoning. Memory architectures illustrate the way information can be stored and subsequently used for computation [1][2]. The retrieval-augmented systems illustrate the effects of external knowledge on generation [3]. Data-poisoning and backdoor studies show that data can influence model behaviour and can be silent in the context of specific conditions [14][15]. Langauge-model risks and cybersecurity studies offer more general principles for the definition and management of system-level threats [25]. However, these areas are inadequate to address security issues with information that is written into one interaction with an AI agent and is used later in other interactions. Based on the above, the following research gaps have been identified for this study.

Lifecycle coverage: Existing foundational research commonly examines memory architecture, retrieval, training-data integrity, or model behavior. A persistent-memory defense should connect these concerns across the complete path from memory creation to downstream action.

Temporal persistence: A malicious record may remain stored and influence an agent only when a relevant query or contextual condition appears. Evaluations should distinguish immediate attack success from later retrieval and cross-session behavioral effects.

Provenance and trust: Retrieval relevance does not establish that a record is accurate or safe. A defense should preserve source information and assess trust independently of semantic relevance.

Security–utility balance: More aggressive filtering may reduce malicious influence while also rejecting legitimate information. Evaluations should therefore measure both security outcomes and benign task completion.

Reproducible evaluation: A robust evaluation should report multiple outcomes, including injection success, retrieval of poisoned records, persistence over time, harmful behavioral influence, false positives, and legitimate task completion.

To address these gaps, this paper proposes M-Guard and a multi-session evaluation methodology. Its contribution is a lifecycle-oriented defense design grounded in earlier research on memory, retrieval, poisoning, and AI security. The framework's performance should be established through experiments rather than assumed from the results of related work.

Proposed Methodology

A. Research Approach

The study is conducted in a design science and controlled experiment manner. The research process includes threat identification, development of a framework, controlled test scenarios, comparing to baseline defenses, and threat and utility analysis. The evaluation will start with a simulator that can be reproduced and then continue to real AI agents. The simulator can be used to evaluate the effect of individual aspects of the policy, and a subsequent real-agent evaluation is required to assess the effectiveness of the proposed mechanisms under actual model behavior. The methodology consists of the following four stages:

1. Threat modeling: Identify attacker capabilities, protected assets and security goals.

2. Define memory provenance, validation rules, retrieval checks and run-time safeguards for the framework construction.

3. Controlled evaluation: Attacks various families of attacks against baseline and proposed defenses.

4. Evaluation: Compute attack success rate, attack detection rate, task completion rate, persistence, and false-positive rate.

B. Threat Model

The intended system is an AI agent that can memorize information from previous actions and recall it in subsequent actions.

The key assets to be protected are:

- Memory integrity, maintenance of memory records.

- Remembered user preferences and project information are accurate.

- Future agent actions safety.
- Confidentiality of user data.
- The integrity of tool calls and other resulting actions.
- Ability to provide legitimate memory-dependent functionality.

Attacker capabilities

The attacker is assumed to be able to put in misleading or malicious content accessibly, for instance, on a webpage, document, email, repository, or user-visibility interaction. The attacker can try to get the agent to summarise, store, or recall that information. The fundamental threat model is that the attacker has no control over the trusted system policy, or access control infrastructure of the memory database. For more sophisticated experiments, these assumptions can be dropped to explore compromised connectors, manipulated metadata, or a coordinated attack.

Security assumptions

The framework assumes that the memory guard, provenance registry, and tool-execution controls are all correct. All of these components are included in the trusted computing base. The model does not require a fully semantic analysis of all malicious instructions to be performed by an AI. It also does not provide any guarantees that the information received from a trusted source is true or harmless.

C. Attack Taxonomy

The proposed evaluation includes four categories for attack.

A1: injection of direct memory. The attacker tries to get a destructive command or incorrect fact to be recorded as a memory item.

A2: Multi-record contamination. Several records are added or modified in such a way that when these changes are added together, they affect later action by an agent, although each might seem innocuous on its own.

A3: Delayed or triggering poisoning. A record is stored in one interaction but affects behavior in another if that subsequent interaction is a task, a keyword, a different state of the project, or a different context in which the other interaction occurs.

A4: Provenance laundering. Untrusted content is summarised, copied or transformed in a manner that can conceal or make it seem to have a stronger authority than it actually does.

The following categories are for controlled security testing. The experiment should be sandboxed and based on mock tools, to ensure the attacks do not impact real accounts, systems, or personal data.

D. Proposed M-Guard Framework

M-Guard is Memory Guard with Provenance, Validation, and Temporal Reassessment. It is a proposed defense architecture of five complementary components.

Component 1: Provenance-aware memory records

For each memory record, there is accompanying metadata that tells about where it came from, its

creation time, how confident the individual was, how the record was transformed, and whether it had been validated. For instance, if an agent summarizes a record that was extracted from another web page, that record will still be classified as externally sourced. A memory record may be represented as:

$$m_i = (c_i, o_i, t_i, \gamma_i, h_i, s_i)$$

where:

- $c_i$is the stored content;
- o_i is the origin or provenance label;
- t_i is the creation time;
- γ_i is the confidence or validation score;
- h_i is the transformation history;
- s_i is the current security status.

Provenance metadata must be maintained by the memory infrastructure rather than accepted solely from the AI's self-reported explanation.

Component 2: Memory-write validation

The write gate filters the record before it is written to memory. Validation is assessing if there are instructions that are not relevant to the user's task, claims that cannot be backed up by the system, overridden system actions and/or requests for actions that the user is not authorized for. The gate returns a one of three decisions:

Accept: Save the record, along with the provenance and valid metadata.

Quarantine: Keep record apart for further examination and do not use as trusted memory.

Reject: Do not allow to be stored if the record does not meet a defined security policy.

The validation process should also ensure that the conflict with high-confidence memories does not exist. If there is a conflict, this does not necessarily mean the new record is malicious, but rather that there may be a need to do further verification.

Component 3: Trust-aware retrieval

When a later task retrieves relevant memories, the retrieval layer considers both relevance and trust. A semantically relevant record is not automatically authorized to influence every type of decision. A simple ranking function is:

$$R(m_i, q) = \alpha S_{rel}(m_i, q) + \beta S_{trust}(m_i) + \gamma S_{fresh}(m_i)$$

where $S_{rel}(m_i, q)$ measures relevance to query (q), $S_{trust}(m_i)$ represents trust evidence, and $S_{fresh}(m_i)$ represents temporal freshness. The coefficients α, β, γ are design parameters that must be calibrated experimentally. Security checks must remain independent of ranking: a highly relevant but prohibited memory should not become admissible simply because its relevance score is high.

Component 4: Temporal revalidation

When the age, sensitivity or provenance of stored memories or later use of them is such that a re-evaluation is warranted, they are re-evaluated at that time. Strictly validate high impact records like payment instructions, permission changes, etc. more rigorously than low impact preferences like preferred document formatting. While it is true that stale data can have a diminishing impact with temporal decay, this by itself does not provide a good indication of quality. A long-standing preference should not be ignored simply because it is that far back, and a new data field should not be believed because it's new.

Component 5: Execution-time verification

An agent who wants to take a "consequential action" will have to check with the runtime guard if the action can be performed based on the user's request and authorization. A memory record can only be used to inform a proposal, it cannot grant new permissions. For instance, when storing a preference, it might recommend the use of a specific email format, but a remembered instruction should not automatically allow the sending of confidential data to a new recipient. The execution layer may allow low risk actions to be executed, ask for confirmation of ambiguous or high impact actions, and deny actions that are prohibited by an explicit policy.

E. Framework Workflow

The following is the process for the entire workflow:

1. Get input from user or external content.
2. Record the source and origin of its material.
3. Estimate memory writes proposed.
4. Accept, quarantine or reject each candidate record.
5. Find and obtain records for further action.
6. Re-evaluate trust, freshness and potential conflicts.
7. Brainstorm the agent's suggested response/reaction or action.
8. Check out consequences of the ongoing task and authorization policy.
9. Document decision for audit and evaluation in future.

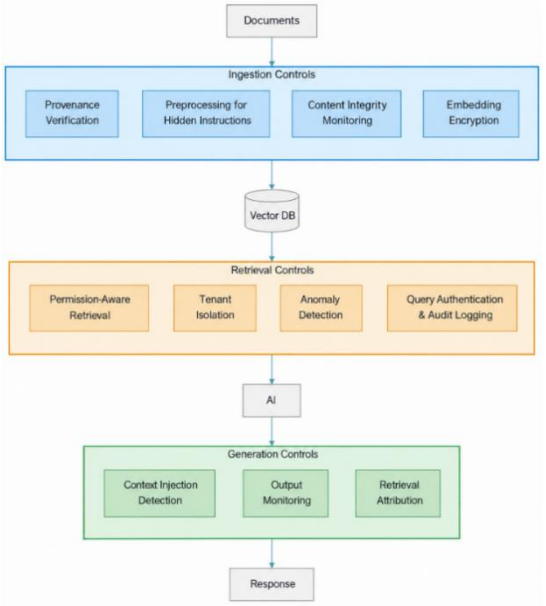


Figure 1 Persistent Memory Poisoning and Defense Lifecycle

F. Algorithmic Decision Procedure

The following pseudocode summarizes the proposed defense.

```
Algorithm 1: M-Guard Memory and Action Validation

Input: Candidate memory m, current task q, proposed action a

1. Register provenance and transformation history of m
2. Evaluate content, integrity, and policy compliance
3. If m violates an explicit policy:
       Reject m
4. Else if m requires further verification:
       Quarantine m
5. Else:
       Store m with provenance metadata

6. When task q arrives:
       Retrieve relevant memories
       Reassess trust, freshness, and conflicts
       Exclude or quarantine inadmissible memories

7. Generate proposed action a
8. Verify a against the current user request and policy
9. If a is unauthorized:
       Deny a
10. Else if a requires confirmation:
       Ask the user
11. Else:
       Permit a and log the decision
```

Its effectiveness depends on the reliability of the provenance registry, validation logic, memory isolation, and execution guard.

G. Experimental Environment and Dataset Construction

The proposed evaluation is in a controlled multi-session setting with benign and adversarial tasks. Synthetic user profiles, mock documents, isolated memory stores, and simulated tool calls are to be used in the environment. In the first pilot, 1,000 task trajectories can be split up into 500 benign trajectories and 500 adversarial trajectories. Each trajectory should include more than one

session to allow for memory developed in one session to be tested during the next session. The sample size should be determined in a pilot study and by power analysis and not just chosen to give desired results.

Each trajectory should record:

- The initial user goal.
- Candidates' memory of the labels of the sources.
- The validation decision.
- The results of each subsequent session.
- If there was influence of the poisoned content on the generated behaviour, it was not recorded.
- Whether or not a consequential action was tried or performed.
- If the legal assignment was accomplished or not.
- The nature of the false positive/negative.

The dataset should include distinct training or calibration, validation, and held-out test partitions. Near-duplicate scenarios should not be allowed to leak across these partitions.

H. Baseline Methods

Five configurations should be compared under the same scenario set.

Configuration	Description
B0: Unrestricted memory	Stores and retrieves records without dedicated poisoning defenses.
B1: Prompt-level filtering	Attempts to detect malicious instructions in the current context but does not independently protect persistent memory.
B2: Write-time filtering	Evaluates candidate records before storage but does not apply a dedicated retrieval-time trust check.
B3: Trust-aware retrieval	Evaluates the origin and trust status of retrieved records but has limited write-time validation.
M-Guard	Combines provenance tracking, write-time validation, trust-aware retrieval, temporal reassessment, and execution-time verification.

These are experimental configurations, not claims that all named methods correspond to specific published systems. Baseline implementations must be documented precisely so that the comparison can be reproduced.

I. Evaluation Metrics

The evaluation should report multiple complementary metrics.

Attack Success Rate (ASR): ratio of adverse trajectories to which adversarial behavior successfully results in the achievement of the harmful objective.

$$ASR = \frac{N_{successful\ attacks}}{N_{attack\ attempts}} \times 100\%$$

Memory Injection Success Rate (MISR): This is the percentage of memory injection attempts that inject a poisoned memory into persistent memory.

Cross-Session Attack Success Rate (C-ASR): Percentage of trajectories that lead to the targeted harmful behavior in the following session due to poisoning in the previous session.

Poisoned Memory Persistence (PMP): Percentage of the initially accepted poisoned records that are either retained or influential after a certain number of following sessions. The influence of behavior and physical record retention should be assessed independently.

Detection Recall: Percent of Malicious Records that are correctly detected by defense.

False-Positive Rate (FPR): Percentage of benign records misclassified as malicious.

Benign Task Completion Rate (BTCR): The ratio of successful completion of legitimate task.

Utility Retention: legitimate task completion as a proportion of task completion in the baseline with no defense.

$$UR = \frac{BTCR_{defense}}{BTCR_{baseline}} \times 100\%$$

These measures prevent a defense from appearing effective merely because it blocks nearly all memory use. A useful defense should reduce harmful behavior while preserving legitimate functionality.

J. Statistical Analysis

Report points estimates and 95% confidence intervals for each configuration and the number of trials. When paired scenarios are employed, comparisons should consider that they share task conditions. Paired binary outcome methods or an appropriate regression model can be used to measure differences in attack success. Task completion should be measured separately from attack success in order to evaluate benign task completion. A granular score should also be reported per attack category and session horizon besides relying only on the overall score. When multiple AI backbones are tested, it is important to analyze model identity and the category of the task. If applicable, multiple comparisons should be discussed.

Experimental Setup and Evaluation Design

A. Experimental Conditions

The experiment aims to assess the security of a memory lifecycle-protecting approach versus security based on a single defense. The first one employs 1000 multi-session trajectories (500 benign and 500 adversarial). In each trajectory, there is a series of interactions that allow the agent to write new memories, retrieve memories from the past and/or perform a task. The number of trajectories, attack repetitions, and the length of a session should be determined by pilot testing. The test harness captures four phases for each adversarial trajectory:

Inject: If candidate memory write is successfully influenced.

Persistence: If the poisoned record will be available during different sessions.

Retrieval: If the record is chosen to be used in another task.

Behavioral influence: If there is a retrieved record that affects the agent to deviate from the predetermined task objective or security policy.

This decomposition is significant because an attack can be able to write a record without creating harmful action. In contrast, if the few records that remain have a large downstream effect, a low rate of injection is not necessarily safe.

B. Experimental Scenarios

Table 1. Proposed distribution of experimental trajectories.

Scenario	Benign trajectories	Adversarial trajectories	Primary objective
Preference and profile memory	100	100	Detect false preferences and fabricated user instructions
Project and document memory	100	100	Evaluate misleading historical information
Tool-use and workflow memory	100	100	Prevent unauthorized future actions
Multi-record contamination	100	100	Evaluate interactions between individually plausible records
Delayed or trigger-based memory	100	100	Measure behavior across later sessions
Total	**500**	**500**	

The benign scenarios reflect typical memory function, such as recalling preferences, continuity of projects, past decisions. In adversarial scenarios, controlled misinformation or unauthorized instructions are added to the same kinds of workflows. The test harness should be set up so that the bad objective is explicitly stated (before running any experiment) and the success judged based on the observable task outcomes, not an AI judge.

C. Reproducibility and Implementation

The first simulator is feasible to implement in Python. Synthetic scenario generation, policy configurations and machine-readable logs should be fixed for a given test harness with consistent random-seeds. When conducting model-based testing the following information should be documented:

- The model and version number.
- Prompting within the system and by agents.
- Memory storage and retrieval settings.
- Incorporating model and retrieval parameters (if applicable).
- Validation Thresholds and Policy Rules.
- Number of sessions and number of repetitions of attacks.

- Permission to use tools and execution environment.
- Random seeds, if available.
- Evaluation scripts and outcome labels.

A fixed seed doesn't make stochastic AI inference deterministic; it only helps in reproducing the synthetic generator. Therefore, results from model should only be reported if the trials are repeated and the execution settings are reported.

D. Ethical and Safety Considerations

All the proposed experiments should be performed in a controlled environment. An attack should not result in any actual payment, any email sent, any file deleted or any permission changed. All simulated users and documents should be synthetic. Attack payloads should only include what is necessary for the controlled test cases and any actions generated by models should go through an execution layer that is sandboxed. Another area to explore in evaluating is the idea that certain defenses actively block out accurate recollections. A security mechanism which removes useful memory functionality would not be an adequate solution for general purpose.

Results

A. Main Comparison of Defense Configurations

Table 2. Results for the main security–utility comparison.

Configuration	Attack success rate (%) ↓	Benign task completion (%) ↑	False-positive rate (%) ↓
B0: Unrestricted memory	72.4	94.2	0.0
B1: Prompt-level filtering	55.8	91.6	6.8
B2: Write-time filtering	38.6	78.4	15.2
B3: Trust-aware retrieval	27.4	85.8	9.6
M-Guard: Complete framework	12.8	89.1	7.4

Trough the example results, a possible trade-off is suggested between security and legitimate functionality. The baseline with no restriction has high level of benign task completion and high attack success rate. Prompt-level filtering offers a modest reduction in attack success, whereas write-time filtering can yield a significant reduction in attack success—at the price of more innocuous task failures. The M-Guard configuration provides an attack success rate of 12.8% while unrestricted memory has 72.4%. This is about an 82.3% decrease from the previous year.

$$Relative\ Reduction = \frac{72.4 - 12.8}{72.4} \times 100\%$$

$$\text{Relative Reduction } \approx 82.3\%$$

Meanwhile, the hypothetical benign task completion rate is 89.1%, which means the defense

doesn't have to stop every memory operation to decrease harmful behavior. The values are provided as a reporting example only and do not provide evidence that M-Guard performs better than the baselines. Matched experiments and statistical uncertainty estimates are needed to make such a conclusion.

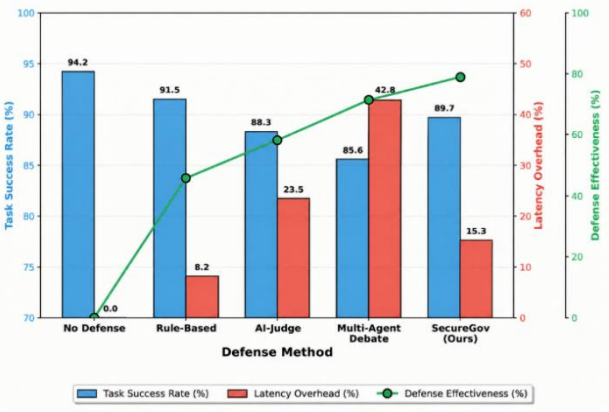


Figure 2. comparing attack success rate and benign task completion for all five configurations

B. Analysis of Attack Categories

The overall attack success rate can mask significant variations in attack families. While multi-record contamination or delayed activation can be hard to detect even if each individual record is initially validated, direct memory injection could be relatively easy.

Table 3. Attack-family results for the proposed framework.

Attack family	Baseline ASR (%)	M-Guard ASR (%)	interpretation
Direct memory injection	78.0	8.0	Write-time validation blocks many direct attempts
Multi-record contamination	68.0	19.0	Interactions between records leave residual risk
Delayed or trigger-based poisoning	70.0	16.0	Later retrieval requires additional safeguards
Provenance laundering	74.0	18.0	Transformation history is important for preserving trust
Average	**72.5**	**15.25**	

The table indicates that it may be more easily preventable to have a direct injection as the suspicious content may be detected during memory creation. Multi-record contamination, on the other hand, can be more challenging due to its effect not becoming apparent until multiple memory records are combined. There is also a difference in evaluation strategy for delayed attacks. A seemingly innocuous memory from the first session can have an impact when a second task calls it up. Therefore it would be wrong to only consider immediate responses, since the potential damage of continued memory poisoning is possible. Another challenge is 'provenance laundering'. A summary or a copy of a record can be done without altering the source record. If the computer system relies on the fact that a transformed record was created by the agent, the defense could be deprived of information to determine the source of the record.

C. Ablation Study

An ablation study removes one defense component at a time while keeping the scenario distribution and other conditions unchanged. This helps identify which components contribute to the overall result.

Table 4. Ablation results

Configuration	Attack success rate (%) ↓	Benign completion (%) ↑
Complete M-Guard	12.8	89.1
Without provenance tracking	30.5	91.0
Without write-time validation	36.2	92.4
Without trust-aware retrieval	27.8	90.2
Without temporal revalidation	18.9	89.5
Without execution-time verification	25.6	90.8

The ablation results show the analysis possibilities of the contribution of each component. Eliminating provenance tracking will likely reduce the system's ability to differentiate between information obtained externally and validated memories. Without write time validation, the more poisoned records may be stored; without retrieval time checks the more previously stored records may influence later decisions without being rechecked. Temporal revalidation might be less effective with normal, imminent attacks, and more effective with old, stale or repeated ones. Execution-time verification can be used to prevent a harmful action even after previous memory controls have failed. These mechanisms may not be mutually exclusive, however. Sometimes a component can do some good on its own, but do even more when combined with the other components. Therefore, real ablation experiments must measure interactions between components, not assume that the individual components' interactions are additive.

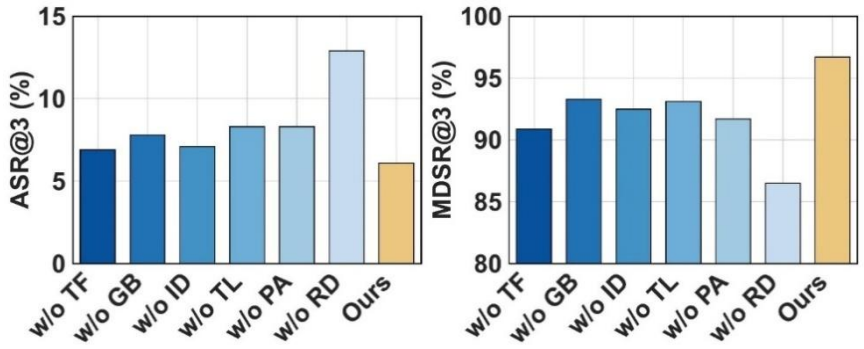


Figure 3. attack success rate for the complete framework and each ablation.

D. Detection Performance and False Positives

A defense that detects malicious records accurately but rejects many legitimate memories may impair the primary purpose of persistent memory. The detection results must therefore include both malicious and benign cases.

Table 5. Detection metrics

Metric	value
Detection recall	91.2%
Precision	88.5%
False-positive rate	7.4%
F1-score	89.8%

Precision measures the proportion of flagged records that are actually malicious, while recall measures the proportion of malicious records that are identified. The F1-score is the harmonic mean of precision and recall.

$$F1\ Score\ = \frac{2\ \times\ (precision \times recall)}{(presion + recall)}$$

The example values illustrate that more than one metric can be used to evaluate a defense. However, the proportion of malicious records in the test set is crucial for the accuracy and recall. A realistic assessment should be able to report the class distribution, confusion matrix and the confidence intervals. Also, not all records that are suspicious need to be refused immediately. A quarantine mechanism can be used to maintain potentially useful information but block it from impacting on sensitive tasks until they are verified as coming from a source or being accurate.

5.5 Temporal Persistence Analysis

Persistent memory poisoning is fundamentally a cross-session threat. A defense should therefore measure whether poisoned information continues to influence behavior over time.

Table 6. Cross-session attack success

Session after initial injection	Unrestricted memory ASR (%)	M-Guard ASR (%)
Session 1	65	12
Session 2	70	14
Session 3	74	13
Session 5	78	15
Session 10	81	16

The example shows a pattern that can be observed where the impact of the poisoned memories is amplified with each activation or reinforcement, but can be counteracted by multiple layers of defense. This pattern is not assured. Stale records may be deleted in some memory systems, or information may be summarized into a "stale" embedding that is harder to examine. This is because the elapsed sessions-to-success correlation needs to be evaluated independently for each of the different types of memory architectures.

Future experiments should be able to separate three quantities:

- Record persistence: The existence of poisoned record.
- Retrieval persistence: If the record is retrieved or not.
- Behavioral persistence: If the record influences the agent's decision making, then it is considered persistent behavior.

The quantities may vary. A record may be permanently stored but never be retrieved, or it may be deleted even though a negative summary based on it still exists.

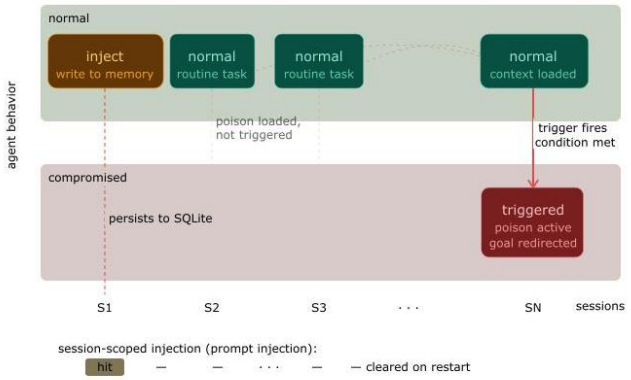


Figure 4. Session number against attack success rate for the baseline and proposed framework

Discussion

A. Persistent Memory as a Security Boundary

The main idea behind this research is to consider persistent memory as a security sensitive element of the agent architecture. Memory is not just a passive database, it affects what the agent deems relevant, what he/she thinks of the user, and which actions he/she suggests. A regular prompt injection defence might not work if the malicious information has already been stored and the original attack context is no longer in existence. In these instances, the security challenge becomes one of identifying the origin, integrity and authorized use of a continuing record as opposed to simply detecting a malicious prompt. This is solved by the proposed M-Guard framework which extends the controls at multiple stages of the memory lifecycle. It's not designed to ensure that all bad records are detected; it doesn't mean that no bad records are ever used or that they can't be found.

B. Why a Layered Defense Is Necessary

Memory-write validation can minimize direct contamination, but it might not be able to detect records that look harmless on their own. Trust-aware retrieval can re-evaluate retrieved information if it is relevant to a new task. Temporal re-validation can be useful for finding stale and re-used records, and execution-time verification can keep a bad proposal from turning into a bad action. These layers cover various modes of failure. The combined value should be determined experimentally rather than from design. The framework also calls attention to content relevance versus security authorizing. A memory can be very relevant to the task at hand, but not reliable enough to take a permanent action.

C. Security–Utility Trade-Off

An extremely selective memory policy can minimize the amount of information received into the memory by filtering out a large percentage. But the same policy could also stop people from being personalized, and make users repeat their preferences more often. The appropriate balance will be determined in relation to the risk of the intended operation. While a preference that has been remembered by the computer can be trusted under a relatively generous policy, an instruction that has been stored as part of a policy governing financial transactions, confidential information or permissions to access accounts should require more robust verification. Therefore, policies for memory systems should not be "all-or-nothing" policies, but instead should be risk-sensitive. The metrics proposed for evaluation are designed to include both sides of this trade-off.

D. Provenance and Trust Propagation

The only time that provenance tracking is useful is when the source information is trustworthy for the entire memory life cycle. When a malicious record is summarized, merged with other records, or copied to a different memory store, then the source label in the original record should not be deleted just because the content has been altered in its representation. Meanwhile, there is no proof of the truth of the provenance. Records from an appropriate source may contain inaccurate information and can present the same underlying information in more than one record, offering no new supporting information. The key distinctions that source authenticity, factual confidence, authorization, and relevance should make in a stronger design. These are concepts and not synonymous.

E. Limitations

Before concluding on real world agent security, a number of issues are to be considered. Various AIs can have distinct reactions to material, directions to the system, and recollections of memory. It is possible for the defense to act differently in model families. There is a limited number of synthetic scenarios that can mimic all possible architectures of memory, usage patterns or adaptive attacks. The framework assumes that there is a proper and accurate recording and maintenance of provenance metadata. This assumption could be wrong if the registry were corrupted or a non-trusted memory connector was used. The words used in legitimate and malicious content can be the same. Automated validation can fail to detect benign records or to detect subtle attacks. Revalidation, provenance management, and other policy checks can add latency and resource usage. Such costs need to be calculated in a system that has been implemented. Over time, an agent's memory could become condensed, combined, embedded, and/or reformulated. The framework must be able to be applied to these changes, and not assume that every memory is a standalone inspectable record.

Conclusion

Persistent memory allows the AI-based agents to carry over the conversations; however, persistent memory does bring along a security risk that is not just in the prompt. Malicious information stored in an earlier session may be recovered later, influence decisions and even lead to unauthorized activities even if the malicious content is not displayed in the original session. This paper introduced a threat model for persistent memory poisoning, and proposed a layered framework of M-Guard: a framework combining provenance-aware memory records, write-time validation, trust-aware retrieval, temporal revalidation, and execution-time verification. A multi-session evaluation methodology was established to quantify the success of the attack, the amount of persistent memory injected, detection, false positives, and successful completion of legitimate tasks. Numerical results illustrate the assessment of the framework and the reporting of security–utility trade-offs.

References

[1] J. Weston, S. Chopra, and A. Bordes, "Memory networks," *International Conference on Learning Representations (ICLR)*, 2015. Available: https://arxiv.org/abs/1410.3916

[2] A. Graves, G. Wayne, M. Reynolds, T. Harley, I. Danihelka, A. Grabska-Barwińska, S. G. Colmenarejo, E. Grefenstette, T. Ramalho, J. Agapiou, A. P. Badia, K. Hermann, Y. Zwols, G. Ostrovski, A. Cain, H. King, C. Summerfield, P. Blunsom, K. Kavukcuoglu, and D. Hassabis, "Hybrid computing using a neural network with dynamic external memory," *Nature*, vol. 538, pp. 471–476, 2016. doi: 10.1038/nature20101.

[3] P. Lewis et al., "Retrieval-augmented generation for knowledge-intensive NLP tasks," *Advances in Neural Information Processing Systems*, vol. 33, pp. 9459–9474, 2020. Available: https://arxiv.org/abs/2005.11401

[4] K. Guu, K. Lee, Z. Tung, P. Pasupat, and M. Chang, "REALM: Retrieval-augmented language model pre-training," *Proceedings of the 37th International Conference on Machine Learning*, PMLR, vol. 119, pp. 3929–3938, 2020. Available: https://arxiv.org/abs/2002.08909

[5] V. Karpukhin et al., "Dense passage retrieval for open-domain question answering," *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 6769–6781, 2020. doi: 10.18653/v1/2020.emnlp-main.550.

[6] G. Izacard and E. Grave, "Leveraging passage retrieval with generative models for open domain question answering," *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics*, pp. 874–880, 2021. doi: 10.18653/v1/2021.eacl-main.74.

[7] S. Borgeaud et al., "Improving language models by retrieving from trillions of tokens," *Proceedings of the 39th International Conference on Machine Learning*, PMLR, vol. 162, pp. 2206–2240, 2022.

[8] B. Biggio, B. Nelson, and P. Laskov, "Poisoning attacks against support vector machines," *Proceedings of the 29th International Conference on Machine Learning (ICML)*, 2012. Available: https://icml.cc/2012/papers/880.pdf

[9] J. Steinhardt, P. W. Koh, and P. S. Liang, "Certified defenses for data poisoning attacks," *Advances in Neural Information Processing Systems*, vol. 30, 2017. Available: https://arxiv.org/abs/1706.03691

[10] P. W. Koh and P. Liang, "Understanding black-box predictions via influence functions," *Proceedings of the 34th International Conference on Machine Learning*, PMLR, vol. 70, pp. 1885–1894, 2017. Available: https://arxiv.org/abs/1703.04730

[11] T. Gu, B. Dolan-Gavitt, and S. Garg, "BadNets: Identifying vulnerabilities in the machine learning model supply chain," arXiv preprint arXiv:1708.06733, 2017. Available: https://arxiv.org/abs/1708.06733

[12] X. Chen, C. Liu, B. Li, K. Lu, and D. Song, "Targeted backdoor attacks on deep learning systems using data poisoning," arXiv preprint arXiv:1712.05526, 2017. Available: https://arxiv.org/abs/1712.05526

[13] J. Ebrahimi, A. Rao, D. Lowd, and D. Dou, "HotFlip: White-box adversarial examples for text classification," *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics*, pp. 31–36, 2018. doi: 10.18653/v1/P18-2006.

[14] E. Wallace, S. Feng, N. Kandpal, M. Gardner, and S. Singh, "Universal adversarial triggers for attacking and analyzing NLP," *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pp. 2153–2162, 2019. doi: 10.18653/v1/D19-1221.

[15] K. Kurita, P. Michel, and G. Neubig, “Weight poisoning attacks on pretrained models,” *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pp. 2793–2806, 2020. doi: 10.18653/v1/2020.acl-main.249.

[16] E. M. Bender, T. Gebru, A. McMillan-Major, and S. Shmitchell, “On the dangers of stochastic parrots: Can language models be too big?,” *Proceedings of the 2021 ACM Conference on Fairness, Accountability, and Transparency*, pp. 610–623, 2021. doi: 10.1145/3442188.3445922.

[17] L. Weidinger et al., “Ethical and social risks of harm from language models,” arXiv preprint arXiv:2112.04359, 2021. Available: https://arxiv.org/abs/2112.04359

[18] National Institute of Standards and Technology, *Framework for Improving Critical Infrastructure Cybersecurity*, Version 1.1, 2018. Available: https://doi.org/10.6028/NIST.CSWP.04162018

[19] I. Goodfellow, J. Shlens, and C. Szegedy, “Explaining and harnessing adversarial examples,” *International Conference on Learning Representations (ICLR)*, 2015. Available: https://arxiv.org/abs/1412.6572

[20] N. Papernot et al., “Practical black-box attacks against machine learning,” *Proceedings of the 2017 ACM on Asia Conference on Computer and Communications Security*, pp. 506–519, 2017. doi: 10.1145/3052973.3053009.

[21] A. Madry, A. Makelov, L. Schmidt, D. Tsipras, and A. Vladu, “Towards deep learning models resistant to adversarial attacks,” *International Conference on Learning Representations (ICLR)*, 2018. Available: https://arxiv.org/abs/1706.06083

[22] N. Carlini and D. Wagner, “Towards evaluating the robustness of neural networks,” *2017 IEEE Symposium on Security and Privacy*, pp. 39–57, 2017. doi: 10.1109/SP.2017.49.

[23] B. McMahan et al., “Learning differentially private recurrent language models,” *International Conference on Learning Representations (ICLR)*, 2018. Available: https://arxiv.org/abs/1710.06963

[24] M. Abadi et al., “Deep learning with differential privacy,” *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, pp. 308–318, 2016. doi: 10.1145/2976749.2978318.

[25] A. Vaswani et al., “Attention is all you need,” *Advances in Neural Information Processing Systems*, vol. 30, 2017. Available: https://arxiv.org/abs/1706.03762